

# Tight Bounds for Active Self-Assembly Using an Insertion Primitive\*

Benjamin Hescott<sup>†</sup>    Caleb Malchik<sup>‡</sup>    Andrew Winslow<sup>§</sup>

## Abstract

We prove two limits on the behavior of a model of self-assembling particles introduced by Dabby and Chen (SODA 2013), called *insertion systems*, where monomers insert themselves into the middle of a growing linear polymer. First, we prove that the expressive power of these systems is equal to context-free grammars, answering a question posed by Dabby and Chen. Second, we prove that systems of  $k$  monomer types can deterministically construct polymers of length  $n = 2^{\Theta(k^{3/2})}$  in  $O(\log^{5/3}(n))$  expected time, and that this is optimal in both the number of monomer types and expected time.

## 1 Introduction

In this work we study a theoretical model of *algorithmic self-assembly*, in which simple particles aggregate in a distributed manner to carry out complex functionality. Perhaps the most well-studied theoretical model of algorithmic self-assembly is the *abstract Tile Assembly Model (aTAM)* of Winfree [22] consisting of square *tiles* irreversibly attach to a growing polyomino-shaped assembly according to matching edge colors. This model is capable of Turing-universal computation [22], self-simulation [8], and efficient assembly of general (scaled) shapes [21] and squares [1, 20]. Despite this power, the model is incapable of assembling shapes efficiently; a single row of  $n$  tiles requires  $n$  tile types and  $\Omega(n^2)$  expected assembly time, and any shape with  $n$  tiles requires  $\Omega(\sqrt{n})$  expected time [1], even if the shape is assembled non-deterministically [3].

Such a limitation may not seem so significant, except that a wide range of biological systems form complex assemblies in time polylogarithmic in the assembly size, as noted in [7, 23]. These biological systems are capable of such growth because their particles (e.g. living cells) *actively* carry out geometric reconfiguration. In the interest of both understanding naturally occurring biological systems and creating synthetic systems with additional capabilities,

---

\*An abstract version of this work has been published as [17].

<sup>†</sup>Tufts University, Department of Computer Science, [hescott@cs.tufts.edu](mailto:hescott@cs.tufts.edu)

<sup>‡</sup>Tufts University, Department of Computer Science, [caleb.malchik@tufts.edu](mailto:caleb.malchik@tufts.edu)

<sup>§</sup>Université Libre de Bruxelles, Département d'Informatique, [awinslow@ulb.ac.be](mailto:awinslow@ulb.ac.be)

several models of *active self-assembly* have been proposed recently. These include the graph grammars of Klavins et al. [14, 15], the *nubots* model of Woods et al. [2, 4, 23], and the insertion systems of Dabby and Chen [7]. Both graph grammars and nubots are capable of a topologically rich set of assemblies and reconfigurations, but rely on stateful particles forming complex bond arrangements. In contrast, insertion systems consist of stateless particles forming a single chain of bonds. Indeed, all insertion systems are captured as a special case of nubots in which a linear polymer is assembled via parallel insertion-like reconfigurations, as in Theorem 5.1 of [24]. The simplicity of insertion systems makes their implementation in matter a more immediately attainable goal; Dabby and Chen [6, 7] describe a direct implementation of these systems in DNA.

We are careful to make a distinction between *active self-assembly*, where assemblies undergo reconfiguration, and *active tile self-assembly* [9, 10, 11, 12, 13, 16, 18, 19], where tile-based assemblies change their bond structure. Active self-assembly enables exponential assembly rates by enabling insertion of new particles throughout the assembly, while active tile self-assembly does not, since the  $\Omega(\sqrt{n})$  expected-time lower bound of Chen and Doty [3] still applies.

## 2 Definitions

Section 2.1 defines standard context-free grammars, as well as a special type called *symbol-pair grammars*, used in Section 3. Section 2.2 defines insertion systems, with a small number of modifications from the definitions given in [7] designed to ease readability. Section 2.3 formalizes the notion of expressive power used in [7].

### 2.1 Grammars

A *context-free grammar*  $\mathcal{G}$  is a 4-tuple  $\mathcal{G} = (\Sigma, \Gamma, \Delta, S)$ . The sets  $\Sigma$  and  $\Gamma$  are the *terminal* and *non-terminal symbols* of the grammar. The set  $\Delta$  consists of *production rules* or simply *rules*, each of the form  $L \rightarrow R_1 R_2 \cdots R_j$  with  $L \in \Gamma$  and  $R_i \in \Sigma \cup \Gamma$ . Finally, the symbol  $S \in \Gamma$  is a special *start symbol*. The *language of*  $\mathcal{G}$ , denoted  $L(\mathcal{G})$ , is the set of finite strings that can be *derived* by starting with  $S$ , and repeatedly replacing a non-terminal symbol found on the left-hand side of some rule in  $\Delta$  with the sequence of symbols on the right-hand side of the rule. The *size* of  $\mathcal{G}$  is  $|\Delta|$ , the number of rules in  $\mathcal{G}$ . If every rule in  $\Delta$  is of the form  $L \rightarrow R_1 R_2$  or  $L \rightarrow t$ , with  $R_1 R_2 \in \Gamma$  and  $t \in \Sigma$ , then the grammar is said to be in *Chomsky normal form*.

A *symbol-pair grammar*, used in Section 3, is a context-free grammar in Chomsky normal form such that each non-terminal symbol is in fact a symbol pair  $(a, d)$ , and each production rule has the form  $(a, d) \rightarrow (a, b)(c, d)$  or  $(a, d) \rightarrow t$ .

## 2.2 Insertion systems

Dabby and Chen [6, 7] describe both a physical implementation and formal model of insertion systems. We briefly review the physical implementation, then give formal definitions.

**Physical implementation.** Short strands of DNA, called *monomers*, are bonded via complementary base sequences to form linear sequences of monomers called *polymers*. Additional monomers are *inserted* into the gap between two adjacent monomers, called an *insertion site*, by bonding to the adjacent monomers and breaking the existing bond between them via a strand displacement reaction (see Figure 1). Each insertion then creates two new insertion sites for additional monomers to be inserted, allowing *construction* of arbitrarily long polymers.

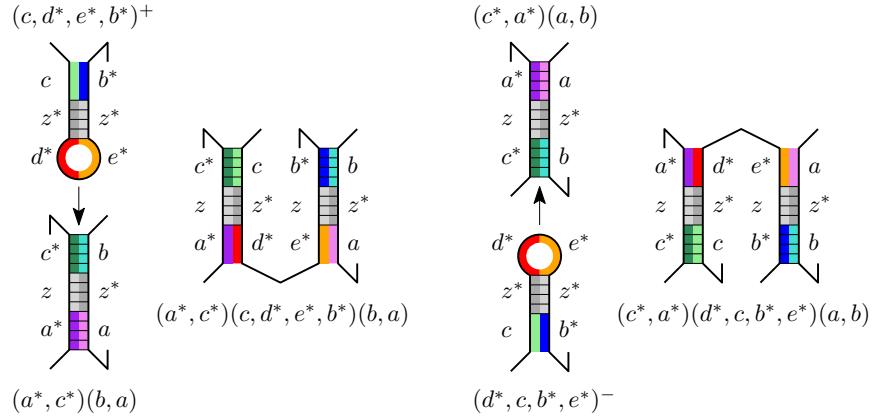


Figure 1: The two types of insertions. Each symbol denotes a DNA subsequence or its complement. The directionality of DNA and hairpin design using generic subsequence symbols  $z, z^*$  creates these distinct types. This figure is loosely based on Figures 2 and 3 of [7].

Each monomer consists of four base sequences that form specific bonds, and only two of these can form bonds during insertion due to the monomer's hairpin design. This design gives each insertion site or monomer one of two *signs* such that a monomer can only be inserted into a site with identical sign.

**Formal model.** An *insertion system*  $\mathcal{S}$  is a 4-tuple  $\mathcal{S} = (\Sigma, \Delta, Q, R)$ . The first element,  $\Sigma$ , is a set of symbols. Each symbol  $s \in \Sigma$  has a *complement*  $s^*$ . We denote the complement of a symbol  $s$  as  $\bar{s}$ , i.e.  $\bar{s} = s^*$  and  $\overline{s^*} = s$ .

The set  $\Delta$  is a set of *monomer types*, each assigned a *concentration*. Each monomer is specified by a signed quadruple  $(a, b, c, d)^+$  or  $(a, b, c, d)^-$ , where  $a, b, c, d \in \Sigma \cup \{s^* : s \in \Sigma\}$ , and is *positive* or *negative* according to its sign. The concentration of each monomer type is a real number between 0 and 1, and the sum of call concentrations is at most 1.

The two symbols  $Q = (a, b)$  and  $R = (c, d)$  are special two-symbol monomers that together form the *initiator* of  $\mathcal{S}$ . It is required that either  $\bar{a} = d$  or  $\bar{b} = c$ .

The *size* of  $\mathcal{S}$  is  $|\Delta|$ , the number of monomer types in  $\mathcal{S}$ .

A *polymer* is a sequence of monomers  $Qm_1m_2\dots m_nR$  where  $m_i \in \Delta$  such that for each pair of adjacent monomers  $(w, x, a, b)(c, d, y, z)$ , either  $\bar{a} = d$  or  $\bar{b} = c$ .<sup>1</sup> The *length* of a polymer is the number of monomers it contains (including  $Q$  and  $R$ ). The gap between every pair of adjacent monomers  $(w, x, a, b)(c, d, y, z)$  in a polymer is an *insertion site*, written  $(a, b)(c, d)$ . Monomers can be *inserted* into an insertion site  $(a, b)(c, d)$  according to the following rules (seen in Figure 1):

1. If  $\bar{a} = d$  and  $\bar{b} \neq c$ , then any monomer  $(\bar{b}, e, f, \bar{c})^+$  can be inserted.
2. If  $\bar{a} \neq d$  and  $\bar{b} = c$ , then any monomer  $(e, \bar{a}, \bar{d}, f)^-$  can be inserted.<sup>2</sup>

A *positive* or *negative* insertion site accepts only positive or negative monomers, respectively. A *dead* insertion site accepts no monomers and has the form  $(a, b)(\bar{b}, \bar{a})$ . An *insertion sequence* is a sequence of insertions, each specified by the site and monomer types, such that each site is created by the previous insertion.

A monomer is inserted after time  $t$ , where  $t$  is an exponential random variable with rate equal to the concentration of the monomer type. The set of all polymers *constructed* by an insertion system is recursively defined as any polymer constructed by inserting a monomer into a polymer constructed by the system, beginning with the initiator. Note that the insertion rules guarantee by induction that for every insertion site  $(a, b)(c, d)$ , either  $\bar{a} = d$  or  $\bar{b} = c$ .

We say that a polymer is *terminal* if no monomer can be inserted into any insertion site in the polymer, and that an insertion system *deterministically constructs* a polymer  $P$  (i.e. is *deterministic*) if every polymer constructed by the system is either  $P$  or is non-terminal and has length less than that of  $P$  (i.e. can become  $P$ ).

The *string representation* of a polymer is the sequence of symbols found on the polymer from left to right, e.g.  $(a, b)(b^*, a, d, c)(c^*, a)$  has string representation  $abb^*adcc^*a$ . We call the set of string representations of all terminal polymers of an insertion system  $\mathcal{S}$  the *language* of  $\mathcal{S}$ , denoted  $L(\mathcal{S})$ .

## 2.3 Expressive power

Intuitively, a system *expresses* another if the terminal polymers or strings created by the system “look” like the terminal polymers or strings created by the other system. In the simplest instance, a symbol-pair grammar  $\mathcal{G}'$  is said to *express* a context-free grammar  $\mathcal{G}$  if  $L(\mathcal{G}') = L(\mathcal{G})$ . Similarly, a grammar  $\mathcal{G}$  is said to *express* an insertion system  $\mathcal{S}$  if  $L(\mathcal{S}) = L(\mathcal{G})$ , i.e. if the set of string representations of the terminal polymers of  $\mathcal{S}$  equals the language of  $\mathcal{G}$ .

<sup>1</sup>For readability, the signs of monomers belonging to a polymer are omitted.

<sup>2</sup>In [7], this rule is described as a monomer  $(\bar{d}, f, e, \bar{a})^-$  that is inserted into the polymer as  $(e, \bar{a}, \bar{d}, f)$ .

An insertion system  $\mathcal{S} = (\Sigma', \Delta', Q', R')$  is said to express a grammar  $\mathcal{G} = (\Sigma, \Gamma, \Delta, S)$  if there exists a function  $g : \Sigma' \cup \{s^* : s \in \Sigma'\} \rightarrow \Sigma \cup \{\varepsilon\}$  and integer  $\kappa$  such that

1.  $\{g(s'_1)g(s'_2)\dots g(s'_n) : s'_1s'_2\dots s'_n \in L(\mathcal{S})\} = L(\mathcal{G})$ .
2. No  $\kappa$  consecutive symbols of a string in  $L(\mathcal{S})$  are mapped to  $\varepsilon$  by  $g$ .

The string representations of polymers have both complementary symbol and length requirements that imply they are unable to capture even simple languages, e.g.  $\{aa\dots a\}$ , despite intuition and claims to the contrary, e.g. Theorem 3.2 of [7] that claims insertion systems express all regular languages. Allowing  $g$  to output  $\varepsilon$  enables locally “cleaning up” string representations to eliminate complementary pairs and other debris, while  $\kappa$  ensures there is a limit on the amount that can be “swept under the rug” locally. A feasible stricter definition could instead use a function  $g : \Delta' \rightarrow \Sigma$  (monomer types of  $\mathcal{S}$  to terminal symbols of  $\mathcal{S}$ ); it is open whether the results presented here would hold under such a definition.

### 3 The Expressive Power of Insertion Systems

Dabby and Chen proved that any insertion system has a context-free grammar expressing it. They construct such a grammar by creating a non-terminal for every possible insertion site and a production rule for every monomer type insertable into the site. For instance, the insertion site  $(a, b)(c^*, a^*)$  and monomer type  $(b^*, d^*, e, c)^+$  induce non-terminal symbol  $A_{(a,b)(c^*,a^*)}$  and production rule  $A_{(a,b)(c^*,a^*)} \rightarrow A_{(a,b)(b^*,d^*)}A_{(e,c)(c^*,a^*)}$ . Here we give a reduction in the other direction, resolving in the affirmative the question posed by Dabby and Chen of whether context-free grammars and insertion systems have the same expressive power:

**Theorem 3.1.** *For every context-free grammar  $G$ , there exists an insertion system that expresses  $G$ .*

The primary difficulty in proving Theorem 3.1 lies in developing a way to simulate the “complete” replacement that occurs during derivation with the “incomplete” replacement that occurs at an insertion site during insertion. For instance,  $bcAbc \Rightarrow bcDDbc$  via a production rule  $A \rightarrow DD$  and  $A$  is completely replaced by  $DD$ . On the other hand, inserting a monomer  $(b^*, d, d, c)^+$  into a site  $(a, b)(c^*, a^*)$  yields the consecutive sites  $(a, b)(b^*, d)$  and  $(d, c)(c^*, a^*)$ , with  $(a, b)(c^*, a^*)$  only partially replaced – the left side of the first site and the right side of second site together form the initial site. This behavior constrains how replacement can be captured by insertion sites, and the  $\kappa$  parameter of the definition of expression (Section 2.3) prevents eliminating the issue via additional insertions.

We overcome this difficulty by proving Theorem 3.1 in two steps. First, we prove that symbol-pair grammars, a constrained type of grammar with incomplete replacements, are able to express context-free grammars (Lemma 3.2).

Second, we prove symbol-pair grammars can be expressed by insertion systems (Lemma 3.3).

**Lemma 3.2.** *For every context-free grammar  $\mathcal{G}$ , there exists a symbol-pair grammar that expresses  $\mathcal{G}$ .*

*Proof.* Let  $\mathcal{G} = (\Sigma, \Gamma, \Delta, S)$ . Let  $n = |\Gamma|$ . Start by putting  $\mathcal{G}$  into Chomsky normal form and then relabeling the non-terminals of  $\mathcal{G}$  to  $A_0, A_1, \dots, A_{n-1}$ , with  $S = A_0$ .

Now we define a symbol-pair grammar  $\mathcal{G}' = (\Sigma', \Gamma', \Delta', S')$  such that  $L(\mathcal{G}') = L(\mathcal{G})$ . Let  $\Sigma' = \Sigma$  and  $\Gamma' = \{(a, d) : 0 \leq a, d < n\}$ ; we treat the symbols in the pairs of  $\Gamma'$  as both symbols and integers.

For each production rule  $A_i \rightarrow A_j A_k$  in  $\Delta$ , add to  $\Delta'$  the set of rules  $(a, d) \rightarrow (a, b)(c, d)$ , with  $0 \leq a < n$ ,  $d = (i - a) \bmod n$ ,  $b = (j - a) \bmod n$ , and  $c = (k - d) \bmod n$ . For each production rule  $A_i \rightarrow t$  in  $\Delta$ , add to  $\Delta'$  the set of rules  $(a, d) \rightarrow t$ , with  $0 \leq a < n$  and  $d = (i - a) \bmod n$ . Let  $S' = (0, 0)$ .

We claim that a partial derivation  $P'$  of  $\mathcal{G}'$  exists if and only if the partial derivation  $P$  obtained by replacing each non-terminal  $(a, d)$  in  $P'$  with  $A_{(a+d) \bmod n}$  is a partial derivation of  $\mathcal{G}$ . By construction, a rule  $(a, d) \rightarrow (a, b)(c, d)$  is in  $\Delta'$  if and only if the rule  $A_{(a+d) \bmod n} \rightarrow A_{(a+b) \bmod n} A_{(c+d) \bmod n}$  is in  $\Delta$ . Similarly, a rule  $(a, d) \rightarrow t$  is in  $\Delta'$  if and only if the rule  $A_{(a+d) \bmod n} \rightarrow t$  is in  $\Delta$ . Also,  $S' = (0, 0)$  and  $S = A_{(0+0) \bmod n}$ . So the claim holds by induction.

Since the set of all partial derivations of  $P'$  are equal to those of  $P$ , the completed derivations are as well and  $L(S') = L(S)$ . So  $\mathcal{G}'$  expresses  $\mathcal{G}$ .  $\square$

**Lemma 3.3.** *For every symbol-pair grammar  $\mathcal{G}$ , there exists an insertion system that expresses  $\mathcal{G}$ .*

*Proof.* Let  $\mathcal{G} = (\Sigma, \Gamma, \Delta, S)$ . The symbol-pair grammar  $\mathcal{G}$  is expressed by an insertion system  $\mathcal{S} = (\Sigma', \Delta', Q', R')$  that we now define. Let  $\Sigma' = \{s_a, s_b : (a, b) \in \Gamma\} \cup \{u, x\} \cup \Sigma$ . Let  $\Delta' = \Delta'_1 \cup \Delta'_2 \cup \Delta'_3 \cup \Delta'_4$ , where

$$\begin{aligned}\Delta'_1 &= \{(s_b, u^*, s_b^*, x)^- : (a, d) \rightarrow (a, b)(c, d) \in \Delta\} \\ \Delta'_2 &= \{(s_a^*, s_b, s_c^*, s_d^*)^+ : (a, d) \rightarrow (a, b)(c, d) \in \Delta\} \\ \Delta'_3 &= \{(x, s_c, u, s_c)^- : (a, d) \rightarrow (a, b)(c, d) \in \Delta\} \\ \Delta'_4 &= \{(s_a^*, t, x, s_d^*)^+ : (a, d) \rightarrow t \in \Delta\}\end{aligned}$$

Let  $Q' = (u, a)$  and  $R' = (b, u^*)$ , where  $S = (a, b)$ .

For instance, the following insertions simulate applying the production rule  $(0, 0) \rightarrow (0, 1)(2, 0)$  to  $(0, 0)$ , where  $\diamond$  denotes the available insertion sites and bold the inserted monomer:

$$\begin{aligned}& (u, s_0) \diamond (s_0, u^*) \\ & (u, s_0) \diamond (\mathbf{s_0^*, s_1, s_2^*, s_0^*}) \diamond (s_0, u^*) \\ & (u, s_0) \diamond (\mathbf{s_1, u^*, s_1^*, x})(s_0^*, s_1, s_2^*, s_0^*) \diamond (s_0, u^*) \\ & (u, s_0) \diamond (s_1, u^*, s_1^*, x)(s_0^*, s_1, s_2^*, s_0^*)(\mathbf{x, s_2, u, s_2}) \diamond (s_0, u^*) \\ & (u, s_0) \diamond (s_1, u^*) \dots (u, s_2) \diamond (s_0, u^*)\end{aligned}$$

The subsequent application of production rules  $(0, 1) \rightarrow p$   $(2, 0) \rightarrow q$  to the string  $(0, 1)(2, 0)$  are simulated by the following insertions:

$$\begin{aligned} & (u, s_0) \diamond (s_1, u^*) \dots (u, s_2) \diamond (s_0, u^*) \\ & (u, s_0)(s_0^*, p, x, s_1^*)(s_1, u^*) \dots (u, s_2) \diamond (s_0, u^*) \\ & (u, s_0)(s_0^*, p, x, s_1^*)(s_1, u^*) \dots (u, s_2)(s_2^*, q, x, s_0^*)(s_0, u^*) \\ & (u, s_0)(s_0^*, p, x, s_1^*) \dots (s_2^*, q, x, s_0^*)(s_0, u^*) \end{aligned}$$

**Insertion types.** First, it is proved that for any polymer constructed by  $\mathcal{S}$ , only three types of insertions of a monomer  $m_2$  between two adjacent monomers  $m_1 m_3$  are possible:

1.  $m_1 \in \Delta'_2, m_2 \in \Delta'_3, m_3 \in \Delta'_1$ .
2.  $m_1 \in \Delta'_3, m_2 \in \Delta'_2 \cup \Delta'_4, m_3 \in \Delta'_1$ .
3.  $m_1 \in \Delta'_3, m_2 \in \Delta'_1, m_3 \in \Delta'_2$ .

Moreover, for every adjacent  $m_1 m_3$  pair satisfying one of these conditions, an insertion of some type  $m_2$  from the specified set is possible.

Consider each possible combination of  $m_1 \in \Delta'_i$  and  $m_3 \in \Delta'_j$ , respectively, with  $i, j \in \{1, 2, 3, 4\}$ . Observe that for an insertion to occur at insertion site  $(a, b)(c, d)$ , the symbols  $\bar{a}, \bar{b}, \bar{c}$ , and  $\bar{d}$  must each occur on some monomer. Then since  $x^*$  and  $t^*$  do not appear on any monomers, any  $i, j$  with  $i \in \{1, 4\}$  or  $j \in \{3, 4\}$  cannot occur. This leaves monomer pairs  $(\Delta'_i, \Delta'_j)$  with  $(i, j) \in \{(2, 1), (2, 2), (3, 1), (3, 2)\}$ .

Insertion sites between  $(\Delta'_2, \Delta'_1)$  pairs have the form  $(s_c^*, s_d^*)(s_d, u^*)$ , so an inserted monomer must have the form  $(\_, s_c, u, \_-)^-$  and is in  $\Delta'_3$ . An insertion site  $(s_c^*, s_d^*)(s_d, u^*)$  implies a rule of the form  $(a, d) \rightarrow (a, b)(c, d)$  in  $\Delta$ , so there exists a monomer  $(x, s_c, u, s_c^*)^- \in \Delta'_3$  that can be inserted.

Insertion sites between  $(\Delta'_3, \Delta'_2)$  pairs have the form  $(u, s_c)(s_c^*, s_b)$ , so an inserted monomer must have the form  $(\_, u^*, s_b^*, \_-)^-$  and thus is in  $\Delta'_1$ . An insertion site  $(u, s_c)(s_c^*, s_b)$  implies a rule of the form  $(c, d) \rightarrow (c, b)(e, d)$  in  $\Gamma$ , so there exists a monomer  $(s_b, u^*, s_b^*, x)^- \in \Delta'_1$  that can be inserted.

Insertion sites between  $(\Delta'_2, \Delta'_2)$  pairs can only occur once a monomer  $m_2 \in \Delta'_2$  has been inserted between a pair of adjacent monomers  $m_1 m_3$  with either  $m_1 \in \Delta'_2$  or  $m_3 \in \Delta'_2$ , but not both. But we just proved that all such possible insertions only permit  $m_2 \in \Delta'_3 \cup \Delta'_1$ . Moreover, the initial insertion site between  $Q'$  and  $R'$  has the form  $(u, s_a)(s_b, u^*)$  of an insertion site with  $m_1 \in \Delta'_3$  and  $m_3 \in \Delta'_1$ . So no pair of adjacent monomers  $m_1 m_3$  are ever both from  $\Delta'_2$  and no insertion site between  $(\Delta'_2, \Delta'_2)$  pairs can ever exist.

Insertion sites between  $(\Delta'_3, \Delta'_1)$  pairs have the form  $(u, s_c)(s_b, u^*)$ , so an inserted monomer must have the form  $(s_c^*, \_, \_, s_b^*)^+$  and is in  $\Delta'_2$  or  $\Delta'_4$ . We prove by induction that for each such insertion site  $(u, s_c)(s_b, u^*)$  that  $(c, b) \in \Gamma$ . First, observe that this is true for the insertion site  $(u, s_a)(s_b, u^*)$  between  $Q'$  and  $R'$ , since  $(a, b) = S \in \Gamma$ . Next, suppose this is true for all insertion sites of some polymer and a monomer  $m_2 \in \Delta'_2 \cup \Delta'_4$  is about to be inserted into the polymer between monomers from  $\Delta'_3$  and  $\Delta'_1$ . Inserting a monomer  $m_2 \in \Delta'_4$

only reduces the set of insertion sites between monomers in  $\Delta'_3$  and  $\Delta'_1$ , and the inductive hypothesis holds. Inserting a monomer  $m_2 \in \Delta'_2$  induces new  $(\Delta'_3, \Delta'_2)$  and  $(\Delta'_2, \Delta'_1)$  insertion site pairs between  $m_1 m_2$  and  $m_2 m_3$ . These pairs must accept two monomers  $m_4 \in \Delta_1$  and  $m_5 \in \Delta_3$ , inducing a sequence of monomers  $m_1 m_4 m_2 m_5 m_3$  with adjacent pairs  $(\Delta'_3, \Delta'_1)$ ,  $(\Delta'_1, \Delta'_2)$ ,  $(\Delta'_2, \Delta'_3)$ ,  $(\Delta'_3, \Delta'_1)$ . Only the first and last pairs permit insertion and both are  $(\Delta'_3, \Delta'_1)$  pairs.

Now consider the details of the three insertions yielding  $m_1 m_4 m_2 m_5 m_3$ , starting with  $m_1 m_3$ . The initial insertion site  $m_1 m_3$  must have the form  $(u, s_a)(s_d, u^*)$ . So the sequence of insertions has the following form, with the last two insertions interchangeable:

$$\begin{aligned} & (u, s_a) \diamond (s_d, u^*) \\ & (u, s_a) \diamond (\mathbf{s}_a^*, \mathbf{s}_b, \mathbf{s}_c^*, \mathbf{s}_d^*) \diamond (s_d, u^*) \\ & (u, s_a) \diamond (\mathbf{s}_b, \mathbf{u}^*, \mathbf{s}_b^*, \mathbf{x})(s_a^*, s_b, s_c^*, s_d^*) \diamond (s_d, u^*) \\ & (u, s_a) \diamond (s_b, u^*, s_b^*, \mathbf{x})(s_a^*, s_b, s_c^*, s_d^*)(\mathbf{x}, \mathbf{s}_c, \mathbf{u}, \mathbf{s}_c) \diamond (s_d, u^*) \end{aligned}$$

Notice the two resulting  $(\Delta'_3, \Delta'_1)$  pair insertion sites  $(u, s_a)(s_b, u^*)$  and  $(u, s_c)(s_d, u^*)$ . Assume, by induction, that the monomer  $m_2$  must exist. So there is a rule  $(a, d) \rightarrow (a, b)(c, d) \in \Delta$  and  $(a, b), (c, d) \in \Gamma$ , fulfilling the inductive hypothesis. So for every insertion site  $(u, s_c)(s_b, u^*)$  between a  $(\Delta'_3, \Delta'_1)$  pair there exists a non-terminal  $(c, b) \in \Gamma$ . So for every adjacent monomer pair  $m_1 m_3$  with  $m_1 \in \Delta'_3$  and  $m_3 \in \Delta'_1$ , there exists a monomer  $m_2 \in \Delta'_2 \cup \Delta'_4$  that can be inserted between  $m_1$  and  $m_2$ .

**Partial derivations and terminal polymers.** Next, consider the sequence of insertion sites between  $(\Delta'_3, \Delta'_1)$  pairs in a polymer constructed by a modified version of  $\mathcal{S}$  lacking the monomers of  $\Delta'_4$ . We claim that a polymer with a sequence  $(u, s_{a_1})(s_{b_1}, u^*), (u, s_{a_2})(s_{b_2}, u^*), \dots, (u, s_{a_i})(s_{b_i}, u^*)$  of  $(\Delta'_3, \Delta'_1)$  insertion sites is constructed if and only if there is a partial derivation  $(a_1, b_1)(a_2, b_2) \dots (a_i, b_i)$  of a string in  $L(\mathcal{G})$ . This follows directly from the previous proof by observing that two new adjacent  $(\Delta'_3, \Delta'_1)$  pair insertion sites  $(u, s_a)(s_b, u^*)$  and  $(u, s_c)(s_d, u^*)$  can replace a  $(\Delta'_3, \Delta'_1)$  pair insertion site if and only if there exists a rule  $(a, d) \rightarrow (a, b)(c, d) \in \Delta$ .

Observe that any string in  $L(\mathcal{G})$  can be derived by first deriving a partial derivation containing only non-terminals, then applying only rules of the form  $(a, d) \rightarrow t$ . Similarly, since the monomers of  $\Delta'_4$  never form half of a valid insertion site, any terminal polymer of  $\mathcal{S}$  can be constructed by first generating a polymer containing only monomers in  $\Delta'_1 \cup \Delta'_2 \cup \Delta'_3$ , then only inserting monomers from  $\Delta'_4$ . Also note that the types of insertions possible in  $\mathcal{S}$  imply that in any terminal polymer, any triple of adjacent monomers  $m_1 m_2 m_3$  with  $m_1 \in \Delta'_i$ ,  $m_2 \in \Delta'_j$ , and  $m_3 \in \Delta'_k$ , that  $(i, j, k) \in \{(4, 1, 2), (1, 2, 3), (2, 3, 4), (3, 4, 1)\}$ , with the first and last monomers of the polymer in  $\Delta'_4$ .

**Expression.** Define the following piecewise function  $g : \Sigma' \cup \{s^* : s \in \Sigma'\} \rightarrow \Sigma \cup \{\varepsilon\}$  that maps to  $\varepsilon$  except for second symbols of monomers in  $\Delta'_4$ .

$$g(s) = \begin{cases} t, & \text{if } t \in \Sigma \\ \varepsilon, & \text{otherwise} \end{cases}$$



Observe that every string in  $L(\mathcal{S})$  has length  $2 + 4 \cdot (4n - 3) + 2 = 16n - 8$  for some  $n \geq 0$ . Also, for each string  $s'_1 s'_2 \dots s'_{16n-8} \in L(\mathcal{S})$ ,  $g(s'_1)g(s'_2) \dots g(s'_{16n-8}) = \varepsilon^3 t_1 \varepsilon^{16} t_2 \varepsilon^{16} \dots t_n \varepsilon^5$ . There is a terminal polymer with string representation in  $L(\mathcal{S})$  yielding the sequence  $s_1 s_2 \dots s_n$  if and only if the polymer can be constructed by first generating a terminal polymer excluding  $\Delta'_4$  monomers with a sequence of  $(\Delta'_3, \Delta'_1)$  insertion pairs  $(a_1, b_1)(a_2, b_2) \dots (a_n, b_n)$  followed by a sequence of insertions of monomers from  $\Delta'_4$  with second symbols  $t_1 t_2 \dots t_n$ . Such a generation is possible if and only if  $(a_1, b_1)(a_2, b_2) \dots (a_n, b_n)$  is a partial derivation of a string in  $L(\mathcal{G})$  and  $(a_1, b_1) \rightarrow t_1, (a_2, b_2) \rightarrow t_2, \dots, (a_n, b_n) \rightarrow t_n \in \Delta$ . So applying the function  $g$  to the string representations of the terminal polymers of  $\mathcal{S}$  gives  $L(\mathcal{G})$ , i.e.  $L(\mathcal{S}) = L(\mathcal{G})$ . Moreover, the second symbol in every fourth monomer in a terminal polymer of  $\mathcal{S}$  maps to a symbol of  $\Sigma$  using  $g$ . So  $\mathcal{S}$  expresses  $\mathcal{G}$  with the function  $g$  and  $\kappa = 16$ .  $\square$

## 4 Positive Results for Polymer Growth

Dabby and Chen also consider the size and speed of constructing finite polymers. They give a construction achieving the following result:

**Theorem 4.1** ([7]). *For any positive integer  $r$ , there exists an insertion system with  $O(r^2)$  monomer types that deterministically constructs a polymer of length  $n = 2^{\Theta(r)}$  in  $O(\log^3 n)$  expected time. Moreover, the expected time has an exponentially decaying tail probability.*

Here we improve on this construction significantly in both polymer length and expected running time. In Section 5, we prove that this construction is the best possible with respect to both the polymer length and construction time.

**Theorem 4.2.** *For any positive integer  $r$ , there exists an insertion system with  $O(r^2)$  monomer types that deterministically constructs a polymer of length  $n = 2^{\Theta(r^3)}$  in  $O(\log^{5/3}(n))$  expected time. Moreover, the expected time has an exponentially decaying tail probability.*

*Proof.* The approach is to implement a three variable counter where each variable ranges over the values 0 to  $r$ , effectively carrying out the execution of a triple for-loop. Insertion sites of the form  $(s_a, s_b)(s_c, s_a^*)$  are used to encode the state of the counter, where  $a$ ,  $b$ , and  $c$  are the variables of the outer, inner, and middle loops, respectively. Three types of variable increments are carried out by the counter:

Inner: If  $b < r$ , then  $(s_a, s_b)(s_c, s_a^*) \rightsquigarrow (s_a, s_{b+1})(s_c, s_a^*)$ .

Middle: If  $b = r$  and  $c < r$ , then  $(s_a, s_b)(s_c, s_a^*) \rightsquigarrow (s_a, s_0)(s_{c+1}, s_a^*)$ .

Outer: If  $b = c = r$  and  $a < r$ , then  $(s_a, s_b)(s_c, s_a^*) \rightsquigarrow (s_{a+1}, s_0)(s_0, s_{a+1}^*)$ .

For  $r = 2$ , these increment types give an insertion sequence of the following form from left to right:

$$\begin{array}{ccc}
(s_0, s_0) & (s_0, s_0^*) & (s_1, s_0) & (s_0, s_1^*) & (s_2, s_0) & (s_0, s_2^*) \\
\downarrow \text{inner} \times 2 & & \downarrow \text{inner} \times 2 & & \downarrow \text{inner} \times 2 & \\
(s_0, s_2) & (s_0, s_0^*) & (s_1, s_2) & (s_0, s_1^*) & (s_2, s_2) & (s_0, s_2^*) \\
\downarrow \text{middle} & & \downarrow \text{middle} & & \downarrow \text{middle} & \\
(s_0, s_0) & (s_1, s_0^*) & (s_1, s_0) & (s_1, s_1^*) & (s_2, s_0) & (s_1, s_2^*) \\
\downarrow \text{inner} \times 2 & & \downarrow \text{inner} \times 2 & & \downarrow \text{inner} \times 2 & \\
(s_0, s_2) & (s_1, s_0^*) & (s_1, s_2) & (s_1, s_1^*) & (s_2, s_2) & (s_1, s_2^*) \\
\downarrow \text{middle} & & \downarrow \text{middle} & & \downarrow \text{middle} & \\
(s_0, s_0) & (s_2, s_0^*) & (s_1, s_0) & (s_2, s_1^*) & (s_2, s_0) & (s_2, s_2^*) \\
\downarrow \text{inner} \times 2 & & \downarrow \text{inner} \times 2 & & \downarrow \text{inner} \times 2 & \\
(s_0, s_2) & (s_2, s_0^*) & (s_1, s_2) & (s_2, s_1^*) & (s_2, s_2) & (s_2, s_2^*) \\
\downarrow \text{outer} & & \downarrow \text{outer} & & & \\
(s_1, s_0) & (s_0, s_1^*) & (s_2, s_0) & (s_0, s_2^*) & & 
\end{array}$$

A site is *modified* by an insertion sequence that yields a new usable site where all other sites created by the insertion sequence are unusable. For instance, we modify a site  $(s_a, \mathbf{s}_b)(s_c, s_a^*)$  to become  $(s_a, \mathbf{s}_d)(s_c, s_a^*)$ , written  $(s_a, s_b)(s_c, s_a^*) \rightsquigarrow (s_a, s_d)(s_c, s_a^*)$ , by adding the monomer types  $(s_b^*, x, u, s_c^*)^+$  and  $(x, u^*, s_a, s_d)^-$  to the system, where  $x$  is a special symbol whose complement is not found on any monomer. These two monomer types cause the following insertion sequence, using  $\diamond$  to indicate the site being modified and the inserted monomer shown in bold:

$$\begin{aligned}
& (s_a, s_b) \diamond (s_c, s_a^*) \\
& (s_a, s_b)(\mathbf{s}_b^*, \mathbf{x}, \mathbf{u}, \mathbf{s}_c^*) \diamond (s_c, s_a^*) \\
& (s_a, s_b)(s_b^*, x, u, s_c^*)(\mathbf{x}, \mathbf{u}^*, \mathbf{s}_a, \mathbf{s}_d) \diamond (s_c, s_a^*)
\end{aligned}$$

We call this simple modification, where a single symbol in the insertion site is replaced with another symbol, a *replacement*. There are four types of replacements, seen in Table 1, that can each be implemented by a pair of corresponding monomers.

Replacement	Monomers
$(s_a, \mathbf{s}_b)(s_c, s_a^*) \rightsquigarrow (s_a, \mathbf{s}_d)(s_c, s_a^*)$	$(s_b^*, x, u, s_c^*)^+, (x, u^*, s_a, s_d)^-$
$(s_a, s_b)(\mathbf{s}_c, s_a^*) \rightsquigarrow (s_a, s_b)(\mathbf{s}_d, s_a^*)$	$(s_b^*, u, x, s_c^*)^+, (s_d, s_a^*, u^*, x)^-$
$(\mathbf{s}_b, s_a)(s_a^*, s_c) \rightsquigarrow (\mathbf{s}_d, s_a)(s_a^*, s_c)$	$(x, s_b^*, s_c^*, u)^-, (u^*, x, s_d, s_a)^+$
$(s_b, s_a)(s_a^*, \mathbf{s}_c) \rightsquigarrow (s_b, s_a)(s_a^*, \mathbf{s}_d)$	$(u, s_b^*, s_c^*, x)^-, (s_a^*, s_d, x, u^*)^+$

Table 1: The four types of replacement steps and monomer pairs that implement them. The symbol  $u$  can be any symbol, and  $x$  is a special symbol whose complement does not appear on any monomer.

Each of the three increment types are implemented using a sequence of site modifications. The resulting triple for-loop carries out a sequence of  $\Theta(r^3)$  insertions to construct a  $\Theta(r^3)$ -length polymer. A  $2^{\Theta(r^3)}$ -length polymer is achieved by simultaneously duplicating each site during each inner increment.

In the remainder of the proof, we detail the implementation of each increment type, starting with the simplest: middle increments.

**Middle increment.** A middle increment of a site  $(s_a, s_b)(s_c, s_a^*)$  occurs when the site has the form  $(s_a, s_r)(s_c, s_a^*)$  with  $0 \leq c < r$ , performing the modification  $(s_a, s_r)(s_c, s_a^*) \rightsquigarrow (s_a, s_0)(s_{c+1}, s_a^*)$ . We implement middle increments using a sequence of three replacements:

$$(s_a, s_r)(s_c, s_a^*) \xrightarrow{1} (s_a, s_r)(s_{f_1(c)}, s_a^*) \xrightarrow{2} (s_a, s_0)(s_{f_1(c)}, s_a^*) \xrightarrow{3} (s_a, s_0)(s_{c+1}, s_a^*)$$

where  $f_i(n) = n + 2ir^2$ . Use of the function  $f$  avoids unintended interactions between monomers, since for any  $n_1, n_2 \in \{0, 1, \dots, r\}$ ,  $f_i(n_1) \neq f_j(n_2)$  for all  $i \neq j$ . Compiling this sequence of replacements into monomer types gives the following monomers:

Step 1:  $(s_r^*, s_{f_2(c)}, x, s_c^*)^+$  and  $(s_{f_1(c)}, s_a^*, s_{f_2(c)}, x)^-$ .

Step 2:  $(s_r^*, x, s_{f_3(c)}, s_{f_1(c)}^*)^+$  and  $(x, s_{f_3(c)}, s_a, s_0)^-$ .

Step 3:  $(s_0^*, s_{f_4(c+1)}, x, s_{f_1(c)}^*)^+$  and  $(s_{c+1}, s_a^*, s_{f_4(c+1)}, x)^-$ .

This set of monomers results in the following sequence of insertions:

$$\begin{aligned} & (s_a, s_r) \diamond (s_c, s_a^*) \\ & (s_a, s_r) \diamond (\mathbf{s}_r^*, \mathbf{s}_{f_2(c)}, \mathbf{x}, \mathbf{s}_c^*)(s_c, s_a^*) \\ & (s_a, s_r) \diamond (\mathbf{s}_{f_1(c)}, \mathbf{s}_a^*, \mathbf{s}_{f_2(c)}, \mathbf{x})(s_r^*, s_{f_2(c)}, x, s_c^*)(s_c, s_a^*) \\ & (s_a, s_r) \diamond (s_{f_1(c)}, s_a^*) \\ & (s_a, s_r)(\mathbf{s}_r^*, \mathbf{x}, \mathbf{s}_{f_3(c)}, \mathbf{s}_{f_1(c)}^*) \diamond (s_{f_1(c)}, s_a^*) \\ & (s_a, s_r)(s_r^*, x, s_{f_3(c)}, s_{f_1(c)}^*)(\mathbf{x}, \mathbf{s}_{f_3(c)}, \mathbf{s}_a, \mathbf{s}_0) \diamond (s_{f_1(c)}, s_a^*) \\ & (s_a, s_0) \diamond (s_{f_1(c)}, s_a^*) \\ & (s_a, s_0) \diamond (\mathbf{s}_0^*, \mathbf{s}_{f_4(c+1)}, \mathbf{x}, \mathbf{s}_{f_1(c)}^*)(s_{f_1(c)}, s_a^*) \\ & (s_a, s_0) \diamond (\mathbf{s}_{c+1}, \mathbf{s}_a^*, \mathbf{s}_{f_4(c+1)}, \mathbf{x})(s_0^*, s_{f_4(c+1)}, x, s_{f_1(c)}^*)(s_{f_1(c)}, s_a^*) \\ & (s_a, s_0) \diamond (s_{c+1}, s_a^*) \end{aligned}$$

Since each inserted monomer has an instance of  $x$ , all other insertion sites created are unusable. This is true of the insertions used for outer increments and duplications as well.

**Outer increment.** An outer increment of the site  $(s_a, s_b)(s_c, s_a^*)$  occurs when the site has the form  $(s_a, s_r)(s_r, s_a^*)$  with  $0 \leq a < r$ . We implement this step using a four-step sequence of three normal replacements and a special quadruple replacement (Step 2):

$$\begin{aligned} & (s_a, s_r)(s_r, s_a^*) \xrightarrow{1} (s_a, s_{f_6(a)}^*)(s_r, s_a^*) \xrightarrow{2} (s_{a+1}, s_{f_7(r)})(s_{f_6(a)}, s_{a+1}^*) \\ & (s_{a+1}, s_{f_7(r)})(s_{f_6(a)}, s_{a+1}^*) \xrightarrow{3} (s_{a+1}, s_0)(s_{f_6(a)}, s_{a+1}^*) \xrightarrow{4} (s_{a+1}, s_0)(s_0, s_{a+1}^*) \end{aligned}$$

As with middle increments, we compile replacement steps 1, 2, and 4 into monomers using Table 1:

Step 1:  $(s_r^*, x, s_{f_5(r)}, s_r^*)^+$  and  $(x, s_{f_5(r)}^*, s_a, s_{f_6(a)}^*)^-$ .

Step 2:  $(s_{f_6(a)}, s_{a+1}^*, x, s_r^*)^+$  and  $(x, s_a^*, s_{a+1}, s_{f_7(r)})^-$ .

Step 3:  $(s_{f_7(r)}^*, x, s_{f_8(r)}, s_{f_6(a)}^*)^+$  and  $(x, s_{f_8(r)}^*, s_{a+1}, s_0)^-$ .

Step 4:  $(s_0^*, s_{f_9(a)}, x, s_{f_6(a)}^*)^+$  and  $(s_0, s_{a+1}^*, s_{f_9(a)}^*, x)^-$ .

Here is the sequence of insertions, using  $\diamond$  to indicate the site being modified and the inserted monomer shown in bold:

$$\begin{aligned}
& (s_a, s_r) \diamond (s_r, s_a^*) \\
& (s_a, s_r) (\mathbf{s_r^*, x, s_{f_5(r)}, s_r^*}) \diamond (s_r, s_a^*) \\
& (s_a, s_r) (s_r^*, x, s_{f_5(r)}, s_r^*) (\mathbf{x, s_{f_5(r)}^*, s_a, s_{f_6(a)}^*}) \diamond (s_r, s_a^*) \\
& (s_a, s_{f_6(a)}^*) \diamond (s_r, s_a^*) \\
& (s_a, s_{f_6(a)}^*) \diamond (\mathbf{s_{f_6(a)}, s_{a+1}^*, x, s_r^*}) (s_r, s_a^*) \\
& (s_a, s_{f_6(a)}^*) (\mathbf{x, s_a^*, s_{a+1}, s_{f_7(r)}}) \diamond (s_{f_6(a)}, s_{a+1}^*, x, s_r^*) (s_r, s_a^*) \\
& (s_{a+1}, s_{f_7(r)}) \diamond (s_{f_6(a)}, s_{a+1}^*) \\
& (s_{a+1}, s_{f_7(r)}) (\mathbf{s_{f_7(r)}^*, x, s_{f_8(r)}, s_{f_6(a)}^*}) \diamond (s_{f_6(a)}, s_{a+1}^*) \\
& (s_{a+1}, s_{f_7(r)}) (s_{f_7(r)}^*, x, s_{f_8(r)}, s_{f_6(a)}^*) (\mathbf{x, s_{f_8(r)}^*, s_{a+1}, s_0}) \diamond (s_{f_6(a)}, s_{a+1}^*) \\
& (s_{a+1}, s_0) \diamond (s_{f_6(a)}, s_{a+1}^*) \\
& (s_{a+1}, s_0) \diamond (\mathbf{s_0^*, s_{f_9(a)}, x, s_{f_6(a)}^*}) (s_{f_6(a)}, s_{a+1}^*) \\
& (s_{a+1}, s_0) \diamond (\mathbf{s_0, s_{a+1}^*, s_{f_9(a)}^*, x}) (s_0^*, s_{f_9(a)}, x, s_{f_6(a)}^*) (s_{f_6(a)}, s_{a+1}^*) \\
& (s_{a+1}, s_0) \diamond (s_0, s_{a+1}^*)
\end{aligned}$$

**Inner increment.** The inner increment has two phases. The first phase (Steps 1-2) performs duplication, modifying the initial site to a pair of sites:  $(s_a, s_b)(s_c, s_a^*) \rightsquigarrow (s_a, s_b)(s_{f_{10}(c)}, s_a^*) \dots (s_a, s_{b+1})(s_c, s_a^*)$ , yielding an incremented version of the original site and one other site. The second phase (Steps 3-5) is  $(s_a, s_b)(s_{f_{10}(c)}, s_a^*) \rightsquigarrow (s_a, s_{b+1})(s_c, a^*)$ , transforming the second site into an incremented version of the original site.

For the first phase, we use the three monomers:

Step 1:  $(s_b^*, s_{f_{10}(c)}, s_{f_{10}(b+1)}, s_c^*)^+$ .

Step 2:  $(s_{f_{11}(c)}, s_a^*, s_{f_{10}(c)}^*, x)^-$  and  $(x, s_{f_{10}(b+1)}^*, s_a, s_{b+1})^-$ .

The resulting sequence of insertions is

$$\begin{aligned}
& (s_a, s_b) \diamond (s_c, s_a^*) \\
& (s_a, s_b) \diamond (\mathbf{s_b^*, s_{f_{10}(c)}, s_{f_{10}(b+1)}, s_c^*}) \diamond (s_c, s_a^*) \\
& (s_a, s_b) \diamond (\mathbf{s_{f_{11}(c)}, s_a^*, s_{f_{10}(c)}^*, x}) (s_b^*, s_{f_{10}(c)}, s_{f_{10}(b+1)}, s_c^*) \diamond (s_c, s_a^*) \\
& (s_a, s_b) \diamond (s_{f_{11}(c)}, s_a^*, s_{f_{10}(c)}^*, x) (s_b^*, s_{f_{10}(c)}, s_{f_{10}(b+1)}, s_c^*) (\mathbf{x, s_{f_{10}(b+1)}^*, s_a, s_{b+1}}) \diamond (s_c, s_a^*) \\
& (s_a, s_b) \diamond (s_{f_{11}(c)}, s_a^*) \dots (s_a, s_{b+1}) \diamond (s_c, s_a^*)
\end{aligned}$$

The last two insertions occur independently and may happen in the opposite order of the sequence depicted here. In the second phase, the site  $(s_a, s_b)(s_{f_{11}(c)}, s_a^*)$  is transformed into  $(s_a, s_{b+1})(s_c, s_a^*)$  by a sequence of replacement steps:

$$(s_a, s_b)(s_{f_{11}(c)}, s_a^*) \xrightarrow{3} (s_a, s_{f_{12}(b)})(s_{f_{11}(c)}, s_a^*) \xrightarrow{4} (s_a, s_{f_{12}(b)})(s_c, s_a^*) \xrightarrow{5} (s_a, s_{b+1})(s_c, s_a^*)$$

As with previous sequences of replacement steps, we compile this sequence into a set of monomers:

$$\text{Step 3: } (s_b^*, x, s_{f_{13}(b)}, s_{f_{11}(c)}^*)^+ \text{ and } (x, s_{f_{13}(b)}^*, s_a, s_{f_{12}(b)})^-.$$

$$\text{Step 4: } (s_{f_{12}(b)}^*, s_{f_{14}(c)}, x, s_{f_{11}(c)}^*)^+ \text{ and } (s_c, s_a^*, s_{f_{14}(c)}^*, x)^-.$$

$$\text{Step 5: } (s_{f_{12}(b)}^*, x, s_{f_{15}(b+1)}, s_c^*)^+ \text{ and } (x, s_{f_{15}(b+1)}^*, s_a, s_{b+1})^-.$$

The resulting sequence of insertions is

$$\begin{aligned} & (s_a, s_b) \diamond (s_{f_{11}(c)}, s_a^*) \\ & (s_a, s_b)(s_b^*, x, s_{f_{13}(b)}, s_{f_{11}(c)}^*) \diamond (s_{f_{11}(c)}, s_a^*) \\ & (s_a, s_b)(s_b^*, x, s_{f_{13}(b)}, s_{f_{11}(c)}^*)(x, s_{f_{13}(b)}^*, s_a, s_{f_{12}(b)}) \diamond (s_{f_{11}(c)}, s_a^*) \\ & (s_a, s_{f_{12}(b)}) \diamond (s_{f_{11}(c)}, s_a^*) \\ & (s_a, s_{f_{12}(b)}) \diamond (s_{f_{12}(b)}^*, s_{f_{14}(c)}, x, s_{f_{11}(c)}^*)(s_{f_{11}(c)}, s_a^*) \\ & (s_a, s_{f_{12}(b)}) \diamond (s_c, s_a^*, s_{f_{14}(c)}^*, x)(s_{f_{12}(b)}^*, s_{f_{14}(c)}, x, s_{f_{11}(c)}^*)(s_{f_{11}(c)}, s_a^*) \\ & (s_a, s_{f_{12}(b)}) \diamond (s_c, s_a^*) \\ & (s_a, s_{f_{12}(b)})(s_{f_{12}(b)}^*, x, s_{f_{15}(b+1)}, s_c^*) \diamond (s_c, s_a^*) \\ & (s_a, s_{f_{12}(b)})(s_{f_{12}(b)}^*, x, s_{f_{15}(b+1)}, s_c^*)(x, s_{f_{15}(b+1)}^*, s_a, s_{b+1}) \diamond (s_c, s_a^*) \\ & (s_a, s_{b+1}) \diamond (s_c, s_a^*) \end{aligned}$$

When combined, the two phases of duplication modify  $(s_a, s_b)(s_c, s_a^*)$  to become  $(s_a, s_{b+1})(s_c, s_a^*) \dots (s_a, s_{b+1})(s_c, s_a^*)$ , where all sites between the duplicated sites are unusable. Notice that although we need to duplicate  $\Theta(r^3)$  distinct sites, only  $\Theta(r^2)$  monomers are used in the implementation since each monomer either does not depend on  $a$ , e.g.  $(s_b^*, x, s_{f_{13}(b)}, s_{f_{11}(c)}^*)^+$ , or does not depend on  $c$ , e.g.  $(x, s_{f_{13}(b)}^*, s_a, s_{f_{12}(b)})^-$ .

**Putting it together.** The system starts with the initiator  $(s_0, s_0)(s_0, s_0^*)$ . Each increment of the counter occurs either through a middle increment, outer increment, or a duplication. The total set of monomers is seen in Table 2. There are at most  $(r+1)^2$  monomer types in each family (each row of Table 2) and  $O(r^2)$  monomer types total.

The system is deterministic if no pair of monomers can be inserted into any insertion site appearing during construction. It can be verified by an inspection of Table 2 that any two positive monomers have distinct pairs of first and fourth symbols, and any pair of negative monomers have distinct pairs of second and

Step	Inner monomer types ( $b < r$ )	
1	$(s_b^*, s_{f_{10}(c)}, s_{f_{10}(b+1)}, s_c^*)^+$	
2	$(s_{f_{11}(c)}, s_a^*, s_{f_{10}(c)}, x)^-$	$(x, s_{f_{10}(b+1)}^*, s_a, s_{b+1})^-$
3	$(s_b^*, x, s_{f_{13}(b)}, s_{f_{11}(c)}^*)^+$	$(x, s_{f_{13}(b)}^*, s_a, s_{f_{12}(b)})^-$
4	$(s_{f_{12}(b)}^*, s_{f_{14}(c)}, x, s_{f_{11}(c)}^*)^+$	$(s_c, s_a^*, s_{f_{14}(c)}^*, x)^-$
5	$(s_{f_{12}(b)}^*, x, s_{f_{15}(b+1)}, s_c^*)^+$	$(x, s_{f_{15}(b+1)}^*, s_a, s_{b+1})^-$
Step	Middle monomer types ( $c < r$ )	
1	$(s_r^*, s_{f_2(c)}, x, s_c^*)^+$	$(s_{f_1(c)}, s_a^*, s_{f_2(c)}^*, x)^-$
2	$(s_r^*, x, s_{f_3(c)}, s_{f_1(c)}^*)^+$	$(x, s_{f_3(c)}^*, s_a, s_0)^-$
3	$(s_0^*, s_{f_4(c+1)}, x, s_{f_1(c)}^*)^+$	$(s_{c+1}, s_a^*, s_{f_4(c+1)}^*, x)^-$
Step	Outer monomer types ( $a < r$ )	
1	$(s_r^*, x, s_{f_5(r)}, s_r^*)^+$	$(x, s_{f_5(r)}^*, s_a, s_{f_6(a)}^*)^-$
2	$(s_{f_6(a)}, s_{a+1}^*, x, s_r^*)^+$	$(x, s_a^*, s_{a+1}, s_{f_7(r)})^-$
3	$(s_{f_7(r)}^*, x, s_{f_8(r)}, s_{f_6(a)}^*)^+$	$(x, s_{f_8(r)}^*, s_{a+1}, s_0)^-$
4	$(s_0^*, s_{f_9(a)}, x, s_{f_6(a)}^*)^+$	$(s_0, s_{a+1}^*, s_{f_9(a)}^*, x)^-$

Table 2: The set of all monomer types used to deterministically construct a monomer of size  $2^{\Theta(r^3)}$  using  $O(r^2)$  monomer types.

third symbols. So no two monomers can be inserted into the same site and thus the system is deterministic.

The size  $P_i$  of a subpolymer with an initiator encoding some value  $i$  between 0 and  $(r+1)^3 - 1$  can be bounded by  $2P_{i+2} + 9 \leq P_i \leq 2P_{i+1} + 9$ , since either  $i+1$  or  $i+2$  is an inner increment step and no step inserts more than 9 monomers. Moreover,  $P_{(r+1)^3-2} \geq 1$ . So  $P_0 + 2$ , the size of the terminal polymer, is  $2^{\Theta(r^3)}$ .

**Running time.** Define the concentration of each monomer type to be equal. There are  $12r^2 + 24r + 3 \leq 39r^2$  monomer types, so each monomer type has concentration at least  $1/(39r^2)$ . The polymer is complete as soon as every counter's variables have reached the value  $a = b = c = r$ , i.e. every site encoding a counter has been modified to become  $(s_r, s_r)(s_r, s_r^*)$  and the monomer  $(s_r^*, x, s_{f_5(r)}, s_r^*)^+$  has been inserted.

There are fewer than  $2^{r^3}$  such insertions, and each insertion requires at most  $9 \cdot (r+1)^3 \leq 72r^3$  previous insertions to occur. So an upper bound on the expected time  $T_r$  for each such insertion is described as a sum of  $72r^3$  random variables, each with expected time  $39r^2$ . The Chernoff bound for independent exponential random variables [5] implies the following upper bound on  $T_r$ :

$$\begin{aligned} \text{Prob}[T_r > 39r^2 \cdot 72r^3(1 + \delta)] &\leq e^{-39 \cdot 72r^5 \delta^2 / (2 + \delta)} \\ &\leq e^{-r^5 \delta^2 / (2 + \delta)} \\ &\leq e^{-r^5 \delta^2 / (2\delta)} \text{ for all } \delta \geq 2 \\ &\leq e^{-r^5 \delta / 2} \end{aligned}$$

Let  $T_{S_r}$  be the total running time of the system. Then we can bound  $T_{S_r}$  from above using the bound for  $T_r$ :

$$\begin{aligned} \text{Prob}[T_{S_r} > 39r^2 \cdot 72r^3(1 + \delta)] &\leq 2^{r^3} \cdot e^{-r^5 \delta / 2} \\ &\leq 2^{r^3} 2^{-r^5 \delta / 2} \\ &\leq 2^{r^3 - r^5 \delta / 2} \\ &\leq 2^{r^5 \delta / 4 - r^5 \delta / 2} \text{ for all } \delta \geq 4 \\ &\leq 2^{-r^5 \delta / 4} \end{aligned}$$

So  $\text{Prob}[T_{S_r} > 39r^2 \cdot 72r^3(1 + \delta)] \leq 2^{-r^5 \delta / 4}$  for all  $\delta \geq 4$ . So the expected value of  $T_{S_r}$ , the construction time, is  $O(r^5) = O(\log^{5/3}(n))$  with an exponentially decaying tail probability.  $\square$

## 5 Negative Results for Polymer Growth

Here we show that the construction in the previous section is the best possible. We start by proving a helpful lemma on the number of insertion sites that accept

at least one monomer type, which we call *usable* insertion sites.

**Lemma 5.1.** *Any insertion system with  $k$  monomer types has at most  $4k^{3/2}$  usable insertion sites.*

*Proof.* Let  $\mathcal{S} = (\Sigma, \Delta, Q, R)$  be an insertion system that deterministically constructs a polymer of length  $n$ . Let  $k = |\Delta|$  (the number of monomer types in  $\mathcal{S}$ ), and relabel the symbols in  $\Sigma \cup \{s^* : s \in \Sigma\}$  as  $s_1, s_2, \dots, s_{4k}$ , with some of these symbols possibly unused. Define the sets  $L_i = \{(s_a, s_b, s_i, s_c)^\pm \in \Delta\}$  and  $R_i = \{(s_a, s_i, s_b, s_c)^\pm \in \Delta\}$ . We will consider the number of usable insertion sites of  $\mathcal{S}$ , and define  $U_i = \{(s_i, s_b)(s_c, \bar{s}_i) \text{ is usable}\}$ .

Since each monomer type can only be inserted into one site in each  $U_i$ ,  $|U_i| \leq k$ , and since each usable site requires a distinct pair of right and left monomer pairs,  $|U_i| \leq |L_i| \cdot |R_i|$ . So  $|U_i| = \min(k, |L_i| \cdot |R_i|)$ . Since each monomer type appears in exactly one  $L_i$  and  $R_i$ ,  $\sum_{i=1}^{4k} |L_i| = \sum_{i=1}^{4k} |R_i| = k$ .

Consider maximizing  $\sum_{i=1}^{4k} |U_i| = \sum_{i=1}^{4k} \min(k, |L_i| \cdot |R_i|)$  subject to  $\sum_{i=1}^{4k} |L_i| = \sum_{i=1}^{4k} |R_i| = k$ . Clearly  $|L_i| \cdot |R_i| \leq \max(|L_i|, |R_i|)^2$ , and if we define  $B_i = L_i \cup R_i$ ,  $|L_i| \cdot |R_i| \leq |B_i|^2$ . Then  $\sum_{i=1}^{4k} |U_i| \leq \sum_{i=1}^{4k} |B_i|^2$  with  $\sum_{i=1}^{4k} |B_i| = 2k$  and  $|B_i| \leq \sqrt{k}$ . So  $\sum_{i=1}^{4k} |U_i| \leq (\sqrt{k})^2 \cdot 2\sqrt{k}$  and thus  $\sum_{i=1}^{4k} |U_i| \leq 2k^{3/2}$ . So the set of all usable sites of the form  $(s_i, s_b)(s_c, \bar{s}_i)$  has size  $2k^{3/2}$ .

A similar argument using the monomer sets  $L'_i = \{(s_a, s_b, s_c, s_i)^\pm \in \Delta\}$ ,  $R'_i = \{(s_i, s_a, s_b, s_c)^\pm \in \Delta\}$ , and insertion site set  $U'_i = \{(s_b, s_i)(\bar{s}_i, s_c) \text{ is usable}\}$  suffices to prove that the set of all usable sites of the form  $(s_b, s_i)(\bar{s}_i, s_c)$  also has size  $2k^{3/2}$ . Since these describe all usable sites,  $\mathcal{S}$  has at most  $4k^{3/2}$  total usable sites.  $\square$

**Theorem 5.2.** *Any polymer deterministically constructed by an insertion system with  $k$  monomer types has length  $2^{O(k^{3/2})}$ .*

*Proof.* Let  $\mathcal{S}$  be a system with  $k$  monomer types that deterministically constructs a polymer. By Lemma 5.1,  $\mathcal{S}$  has  $O(k^{3/2})$  usable sites. As observed by Dabby and Chen,  $\mathcal{S}$  can be expressed by a grammar  $\mathcal{G}_\mathcal{S}$  with at most  $4k^{3/2}$  non-terminal symbols, where each insertion site  $(a, b)(c, d)$  corresponds to a non-terminal  $A_{a,b,c,d}$ , and each monomer type  $(e, f, g, h)^\pm$  insertable into the site corresponds to a rule  $A_{a,b,c,d} \rightarrow A_{a,b,e,f} A_{g,h,c,d}$ .

Let  $\sigma$  be a string in  $L(\mathcal{G}_\mathcal{S})$  of length  $n$ . So the (binary) derivation tree of any derivation of  $\sigma$  contains a path of length at least  $\log_2 n$ . If  $\log_2 n > 4k^{3/2}$ , then this path must contain at least two occurrences of the same non-terminal symbol. The portion of the path between these two occurrences can be pumped to derive strings of arbitrary lengths, so  $L(\mathcal{G}_\mathcal{S})$  is infinite. So  $L(\mathcal{S}) \neq L(\mathcal{G}_\mathcal{S})$  and  $\mathcal{G}_\mathcal{S}$  does not express  $\mathcal{S}$ , a contradiction. Thus  $\log_2 n \leq 4k^{3/2}$  for every string in  $L(\mathcal{G}_\mathcal{S})$  and the length of the polymer deterministically constructed by  $\mathcal{S}$  is  $2^{O(k^{3/2})}$ .  $\square$

**Theorem 5.3.** *Deterministically constructing a polymer of length  $n$  takes  $\Omega(\log^{5/3}(n))$  expected time.*



*Proof.* The proof approach is to prove a lower bound on the expected time to carry out an insertion sequence of length  $\Omega(\log n)$  involving (by Lemma 5.1),  $\Omega(\log n)$  distinct monomer types. This is converted into a minimization problem for the expected time, whose optimal solutions shown algebraically to be  $\Omega(\log^{5/3}(n))$ .

**A long insertion sequence.** Since each insertion only increases the number of insertion sites by one, the system must carry out an insertion sequence of length at least  $\log_2 n$  when constructing the polymer. No insertion site appears twice in this sequence, since otherwise the system (non-deterministically) constructs polymers of arbitrary length.

Suppose, for the sake of contradiction, that an insertion site in the sequence accepts monomer types  $m_1$  and  $m_2$ , and inserts  $m_1$  into some polymer. Then all polymers constructed by the system without  $m_1$  and, separately, the system without  $m_2$  are constructed by the system and each has polymers not constructed by the other. So the system cannot deterministically construct a polymer, a contradiction, and so no insertion site in the sequence accepts more than one monomer type.

Thus the  $\log_2 n$  (or more) distinct insertion sites appearing in the insertion sequence each accept a unique monomer type. The remainder of the proof is develop a lower bound for the total expected time of the insertions in this sequence.

**An optimization problem.** By linearity of expectation, the total expected time of the insertions is equal to the sum of the expected time for each insertion. Because each insertion site accepts a unique monomer type, the expected time to carry out the insertion is equal to the reciprocal of concentration of this type. Let  $k$  be the number of monomer types inserted into the sites in the subsequence. Let  $c_1, c_2, \dots, c_k$  be the sums of the concentrations of these types, and  $x_1, x_2, \dots, x_k$  be the number of times a monomer from each part is inserted during the subsequence. Then the total expected time for all of the insertions in the subsequence is  $\sum_{i=1}^k x_i/c_i$ . Moreover, these variables are subject to the following constraints:

1.  $\sum_{i=1}^k x_i \geq \log_2 n/2$  (total number of insertions is at least  $\log_2 n/2$ ).
2.  $\sum_{i=1}^k c_i \leq 1$  (total concentration is at most 1).
3.  $k \geq \log^{2/3}(n)/4$  (monomer types is at least  $\log^{2/3}(n)/4$ , Lemma 5.1).

**Minimizing expected time.** Consider minimizing the total expected time subject to these constraints, starting with proving that  $x_i/c_i = x_j/c_j$  for all  $1 \leq i, j \leq k$ . That is, that the ratio of the number of times a monomer type is inserted in the subsequence to the type's concentration is equal for all types. Assume, without loss of generality, that  $x_i/c_i > x_j/c_j$  and  $c_i, c_j > 0$ . Then it can be shown algebraically that the following two statements hold:

1. If  $c_j \geq c_i$ , then for sufficiently small  $\varepsilon > 0$ ,  $\frac{x_i}{c_i} + \frac{x_j}{c_j} > \frac{x_i}{c_i + \varepsilon} + \frac{x_j}{c_j - \varepsilon}$ .

2. If  $c_j < c_i$ , then for sufficiently small  $\varepsilon > 0$ ,  $\frac{x_i}{c_i} + \frac{x_j}{c_j} > \frac{x_i}{c_i - \varepsilon} + \frac{x_j}{c_j + \varepsilon}$ .

Since the ratios of every pair of monomer types are equal,

$$\frac{c_i}{1} \leq \frac{c_i}{\sum_{i=1}^k c_i} = \frac{x_i}{\sum_{i=1}^k x_i} \leq \frac{x_i}{\log n}$$

So  $\log n \leq x_i/c_i$  and  $k \log n \leq \sum_{i=1}^k x_i/c_i$ . By Lemma 5.1, since the insertion subsequence has length  $\log(n)/2$  and no repeated insertion sites,  $k \geq \log^{2/3}(n)/4$ . So the total expected time is  $k \log n \geq \log^{2/3}(n)/8$ .  $\square$

## Acknowledgments

The authors thank anonymous reviewers for comments that improved the readability and correctness of the paper.

## References

- [1] L. Adleman, Q. Cheng, A. Goel, and M.-D. Huang. Running time and program size for self-assembled squares. In *Proceedings of 33rd ACM Symposium on Theory of Computing (STOC)*, 2001.
- [2] H.-L. Chen, D. Doty, D. Holden, C. Thachuk, D. Woods, and C.-T. Yang. Fast algorithmic self-assembly of simple shapes using random agitation. In S. Murata and S. Kobayashi, editors, *DNA Computing and Molecular Programming*, volume 8727 of *LNCS*, pages 20–36. Springer Berlin Heidelberg, 2014.
- [3] H.L. Chen and D. Doty. Parallelism and time in hierarchical self-assembly. In *Proceedings of 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1163–1182, 2012.
- [4] M. Chen, D. Xin, and D. Woods. Parallel computation using active self-assembly. In D. Soloveichik and B. Yurke, editors, *DNA Computing and Molecular Programming*, volume 8141 of *LNCS*, pages 16–30. Springer Berlin Heidelberg, 2013.
- [5] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.
- [6] N. Dabby. *Synthetic molecular machines for active self-assembly : prototype algorithms, designs, and experimental study*. PhD thesis, Caltech, 2013.
- [7] N. Dabby and H.-L. Chen. Active self-assembly of simple units using an insertion primitive. In *Proceedings of 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1526–1536, 2013.

- [8] D. Doty, J. H. Lutz, M. J. Patitz, R. T. Schweller, S. M. Summers, and D. Woods. The tile assembly model is intrinsically universal. In *Proceedings of 53rd IEEE Symposium on Foundations of Computer Sciences (FOCS)*, pages 302–310, 2012.
- [9] V. K. Gautam, P. C. Haddow, and M. Kuiper. Reliable self-assembly by self-triggered activation of enveloped DNA tiles. In A.-H. Dediu, C. Martín-Vide, B. Truthe, and M. A. Vega-Rodríguez, editors, *Theory and Practice of Natural Computing*, volume 8273 of *LNCS*, pages 68–79. Springer Berlin Heidelberg, 2013.
- [10] J. Hendricks, J. E. Padilla, M. J. Patitz, and T. A. Rogers. Signal transmission across tile assemblies: 3D static tiles simulate active self-assembly by 2D signal-passing tiles. In D. Soloveichik and B. Yurke, editors, *DNA Computing and Molecular Programming*, volume 8141 of *LNCS*, pages 90–104. Springer Berlin Heidelberg, 2013.
- [11] N. Jonoska and D. Karpenko. Active tile self-assembly, part 1: universality at temperature 1. *International Journal of Foundations of Computer Science*, 25(2):141–163, 2014.
- [12] N. Jonoska and D. Karpenko. Active tile self-assembly, part 2: self-similar structures and structural recursion. *International Journal of Foundations of Computer Science*, 25(2):165–194, 2014.
- [13] A. Keenan, R. Schweller, and X. Zhong. Exponential replication of patterns in the signal tile assembly model. In D. Soloveichik and B. Yurke, editors, *DNA Computing and Molecular Programming*, volume 8141 of *LNCS*, pages 118–132. Springer Berlin Heidelberg, 2013.
- [14] E. Klavins. Universal self-replication using graph grammars. In *Proceedings of International Conference on MEMS, NANO, and Smart Systems*, pages 198–204, 2004.
- [15] E. Klavins, R. Ghrist, and D. Lipsky. Graph grammars for self assembling robotic systems. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 5, pages 5293–5300, 2004.
- [16] U. Majumder, T. H. LaBean, and J. H. Reif. Activatable tiles: Compact, robust programmable assembly and other applications. In M. H. Garzon and H. Yan, editors, *DNA Computing and Molecular Programming*, volume 4848 of *LNCS*, pages 15–25. Springer Berlin Heidelberg, 2008.
- [17] C. Malchik and A. Winslow. Tight bounds for active self-assembly using an insertion primitive. In *Proceedings of 22nd European Symposium on Algorithms (ESA)*, pages 677–688, 2014.
- [18] J. E. Padilla, W. Liu, and N. C. Seeman. Hierarchical self assembly of patterns from the robinson tilings: DNA tile design in an enhanced tile assembly model. *Natural Computing*, 11(2):323–338, 2012.

- [19] J. E. Padilla, M. J. Patitz, R. T. Schweller, N. C. Seeman, S. M. Summers, and X. Zhong. Asynchronous signal passing for tile self-assembly: fuel efficient computation and efficient assembly of shapes. *International Journal of Foundations of Computer Science*, 25(4):459–488, 2014.
- [20] P. W. K. Rothmund and E. Winfree. The program-size complexity of self-assembled squares (extended abstract). In *Proceedings of 32nd ACM Symposium on Theory of Computing (STOC)*, pages 459–468, 2000.
- [21] D. Soloveichik and E. Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007.
- [22] E. Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, Caltech, 1998.
- [23] D. Woods, H.-L. Chen, S. Goodfriend, N. Dabby, E. Winfree, and P. Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Proceedings of 4th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 353–354, 2013.
- [24] D. Woods, H.-L. Chen, S. Goodfriend, N. Dabby, E. Winfree, and P. Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. Technical report, arXiv, 2013.